A Heterogeneous High Dimensional Approximate Nearest Neighbor Algorithm

Moshe Dubiner

ACKNOWLEDGMENT

I would like to thank Phil Long and David Pablo Cohn for reviewing rough drafts of this paper and suggesting many clarifications. The remaining obscurity is my fault.

M. Dubiner is with Google, e-mail: moshe@google.com

Manuscript submitted to IEEE Transactions on Information Theory on March 3, 2007.

A Heterogeneous High Dimensional Approximate Nearest Neighbor Algorithm

Abstract

We consider the problem of finding high dimensional approximate nearest neighbors. Suppose there are d independent rare features, each having its own independent statistics. A point x will have $x_i = 0$ denote the absence of feature i, and $x_i = 1$ its existence. Sparsity means that usually $x_i = 0$. Distance between points is a variant of the Hamming distance. Dimensional reduction converts the sparse heterogeneous problem into a lower dimensional full homogeneous problem. However we will see that the converted problem can be much harder to solve than the original problem. Instead we suggest a direct approach. It consists of T tries. In try t we rearrange the coordinates in decreasing order of

$$(1 - r_{t,i})\frac{p_{i,11}}{p_{i,01} + p_{i,10}} \ln \frac{1}{p_{i,1*}} \tag{1}$$

where $0 < r_{t,i} < 1$ are uniform pseudo-random numbers, and the $p's$ are the coordinate's statistical parameters. The points are lexicographically ordered, and each is compared to its neighbors in that order.

We analyze a generalization of this algorithm, show that it is optimal in some class of algorithms, and estimate the necessary number of tries to success. It is governed by an information like function, which we call bucketing forest information. Any doubts whether it is "information" are dispelled by another paper, where unrestricted bucketing information is defined.

I. Introduction

Suppose we have two bags of points, X_0 and X_1, randomly distributed in a high-dimensional space. The points are independent of each other, with one exception: there is one unknown point x_0 in bag X_0 that is significantly closer to an unknown point x_1 in bag X_1 than would be accounted for by chance. We want an efficient algorithm for quickly finding these two 'paired' points.

The reader might wonder why we need two sets, instead of working as usual with $X = X_0 \cup X_1$. We have come a full circle on this issue. The practical problem that got us interested in this theory involved texts from two languages, hence two different sets. However it seemed that the asymmetry between X_0 and X_1 was not important, so we developed a one set theory. Than we found out that keeping X_0, X_1 separate makes thing clearer.

Let us start with the well known simple homogeneous marginally Bernoulli(1/2) example. Suppose $X_0, X_1 \subset \{0,1\}^d$ of sizes n_0, n_1 respectively are randomly chosen as independent Bernoulli(1/2) variables, with one exception. Choose randomly one point $x_0 \in X_0$, xor it with a random Bernoulli(p) vector and overwrite one randomly chosen $x_1 \in X_1$. A symmetric description is to say that x_0, x_1 i'th bits have the joint probability

$$P = \begin{pmatrix} p/2 & (1-p)/2 \\ (1-p)/2 & p/2 \end{pmatrix} \tag{2}$$

For some $p > 1/2$. We assume that we know p. In practice it will have to be estimated.

Let

$$\ln M = \ln n_0 + \ln n_1 - I(P)d \tag{3}$$

where

$$I(P) = p\ln(2p) + (1-p)\ln(2(1-p)) \tag{4}$$

is the mutual information between the special pair's single coordinate values. Information theory tells us that we can not hope to pin the special pair down into less than W possibilities, but can come close to it in some asymptotic sense. Assume that W is small. How can we find the closest pair? The trivial way to do it is to compare all the $n_0 n_1$ pairs. A better way has been known for a long time. The earliest references I am aware of are Karp,Waarts and Zweig [6], Broder [3], Indyk and Motwani [5]. They do not limit themselves to this simplistic problem, but their approach clearly handles it. Without restricting generality let $n_0 \leq n_1$. We randomly choose

$$k \approx \log_2 n_0 \tag{5}$$

out of the d coordinates, and compare the point pairs which agree on these coordinates (in other words, fall into the same bucket). The expected number of comparisons is

$$n_0 n_1 2^{-k} \approx n_1 \tag{6}$$

while the probability of success of one comparison is p^k. In case of failure we try again, with other random k coordinates. At first glance it might seem that the expected number of tries until success is p^{-k}, but that is not true because the attempts are interdependent. The correct computation is done in the next section. In the unlimited data case $d \to \infty$ indeed

$$T \approx p^{-k} \approx n_0^{\log_2 1/p} \tag{7}$$

Is this optimal? Alon [1] has suggested the possibility of improvement by using Hamming's perfect code. We have found that in the $n_0 = n_1 = n$ case, $T \approx n^{\log_2 1/p}$ can be reduced to

$$T \approx n^{1/p-1+\epsilon} \tag{8}$$

for any $\epsilon > 0$, see [7]. Unfortunately this seems hard to convert into a practical algorithm.

In practice, most approximate nearest neighbor problems are heterogeneous. Coordinates are not independent either, but there is a lot to learn from the independent case. For starters let the joint probability matrix be position dependent:

$$P_i = \begin{pmatrix} p_i/2 & (1-p_i)/2 \\ (1-p_i)/2 & p_i/2 \end{pmatrix} \qquad 1 \le i \le d \tag{9}$$

This is an important example which we will refer to as the marginally Bernoulli(1/2) example. It turns out that in each try coordinate i should be chosen with probability

$$\max\left[\frac{p_i - p_{cut}}{1 - p_{cut}}, 0\right] \tag{10}$$

for some **cutoff probability** $0 \le p_{cut} \le 1$. An intuitive argument leading to that equation appears in section III.

Section V presents an independent data model, and a general nearest neighbor algorithm using its parameters. Section XIII proves a lower bound for its success probability. Section XII proves an upper bound for a much lager class of algorithms. The lower and upper bound are asymptotically similar. The number of tries T satisfies

$$\ln T \sim \max_{\lambda \ge 0}\left[\lambda \ln n_0 - \sum_{i=1}^{d} F(P_i, \lambda)\right] \tag{11}$$

where $F(P_i, \lambda)$ is defined in (49). The similarity to the information theoretic (3) suggests that $F(P_i, \lambda)$ is some sort of information function. We call it the **bucketing forest information** function. [7] proves a similar estimate for the performance of the "best possible" bucketing algorithm, involving a **bucketing information** function with a very information theoretic look.

Section VIII shows that our algorithm preserves sparseness. Section IX shows that dimensional reduction is bad for sparse data.

II. The Homogeneous Marginally Bernoulli(1/2) Example

The well known homogeneous marginally Bernoulli(1/2) example has been presented in the introduction. It will be analyzed in detail because the main purpose of this paper is generalizing it. The analysis is non-generalizable, but the issues remain. Recall that we have a joint probability matrix

$$P = \begin{pmatrix} p/2 & (1-p)/2 \\ (1-p)/2 & p/2 \end{pmatrix} \tag{12}$$

For some $p > 1/2$. Without restricting generality let $n_0 \leq n_1$. We randomly choose

$$k \approx \min[\log_2 n_0, (2p-1)d] \tag{13}$$

out of the d coordinates. The reason for $k \leq (2p-1)d$ (which was omitted in the introduction for simplicity) will emerge later. We compare point pairs which agree on the chosen k coordinates. This is a random algorithm solving a random problem, so we have two levels of randomness. Usually when we will compute probabilities or expectations it will be with respect to these two sources together. The expected number of comparisons is $n_0 n_1 2^{-k}$ while the probability of success of one comparison is p^k. (These statements are true assuming only model randomness). In case of failure we try again, with other random k coordinates. In order to estimate the expected number of tries till success we have to enumerate how many bits are identical in the special pair x_0, x_1. Let this number be j. Then the probability of success in a single try conditioned on j is $\begin{pmatrix} j \\ k \end{pmatrix} \Big/ \begin{pmatrix} d \\ k \end{pmatrix}$. Hence the expected number of comparisons is Tn_1 where

$$T = n_0 2^{-k} \sum_{j=0}^{d} \begin{pmatrix} d \\ j \end{pmatrix} p^j (1-p)^{d-j} \begin{pmatrix} d \\ k \end{pmatrix} \Big/ \begin{pmatrix} j \\ k \end{pmatrix}$$

For small d/k this is too pessimistic because most of the contribution to the above sum comes from unlikely low j's. We know that with probability about 1/2, $j \geq pd$. Hence we get a success probability of about 1/2 with an expected

$$T = n_0 2^{-k} \sum_{j=pd}^{d} \begin{pmatrix} d \\ j \end{pmatrix} p^j (1-p)^{d-j} \begin{pmatrix} d \\ k \end{pmatrix} \Big/ \begin{pmatrix} j \\ k \end{pmatrix} \approx$$

$$\approx n_0 2^{-k} \begin{pmatrix} d \\ k \end{pmatrix} \Big/ \begin{pmatrix} pd \\ k \end{pmatrix} = n_0 \prod_{i=0}^{k-1} \frac{1 - i/d}{2(p - i/d)}$$

Now it is clear that increasing k above $(2p-1)d$ increases T, which is counterproductive.

III. An Intuitive Argument for the Marginally Bernoulli(1/2) Example

In full generality our algorithm is not very intuitive. In this section we will present an intuitive argument for the special case of the joint probability matrices

$$P_i = \begin{pmatrix} p_i/2 & (1-p_i)/2 \\ (1-p_i)/2 & p_i/2 \end{pmatrix} \qquad 1 \leq i \leq d \tag{14}$$

The impatient reader may skip this and the next section, jumping directly to the algorithm. Let us order the coordinates in decreasing order of importance

$$p_1 \geq p_2 \geq \cdots \geq p_d \tag{15}$$

Moreover let us bunch coordinates together into g groups of $d_1, d_2, \ldots, d_g$ coordinates, where $\sum_{h=1}^{g} d_h = d$, and the members of group h all have the same probability q_h

$$p_{d_1+\cdots+d_{h-1}+1} = \cdots = p_{d_1+\cdots+d_h} = q_h \tag{16}$$

Out of the d_h coordinates in group h, the special pair will agree in approximately $q_h d_h$ 'good' coordinates. Let us make things simple by pretending that this is the exact value (never mind that it is not an integer). We want to choose

$$k = \log_2 n_0 \tag{17}$$

coordinates and compare pairs which agree on them. The greedy approach seems to choose as many as possible from the group 1, but conditional greed disagrees. Let us pick the first coordinate randomly from group 1. If it is bad, the whole try is lost. If it is good, group 1 is reduced to size $d_1 - 1$, out of which $q_1 d_1 - 1$ are good. Hence the probability that a remaining coordinate is good is reduced to

$$\frac{q_1 d_1 - 1}{d_1 - 1} \tag{18}$$

After taking m coordinates out of group 1, its probability decreases to

$$\frac{q_1 d_1 - m}{d_1 - m} \tag{19}$$

Hence after taking

$$m = \frac{q_1 - q_2}{1 - q_2} d_1 \tag{20}$$

coordinates, group 1 merges with group 2. We will randomly chose coordinates from this merged group till its probability drops to q_3. At that point the probability of a second group coordinate to be chosen is

$$\frac{q_2 - q_3}{1 - q_3} \tag{21}$$

while the probability of a first group coordinate being picked either before or after the union is

$$\frac{q_1 - q_2}{1 - q_2} + \left(1 - \frac{q_1 - q_2}{1 - q_2}\right) \frac{q_2 - q_3}{1 - q_3} = \frac{q_1 - q_3}{1 - q_3} \tag{22}$$

This goes on till at some $q_l = p_{cut}$ we have k coordinates. Then the probability that coordinate i is chosen is

$$\max\left[\frac{p_i - p_{cut}}{1 - p_{cut}}, 0\right] \tag{23}$$

as stated in the introduction. The cutoff probability is determined by

$$\sum_{i=1}^{d} \max\left[\frac{p_i - p_{cut}}{1 - p_{cut}}, 0\right] \approx k \tag{24}$$

The previous equation can be iteratively solved. However it is better to look from a different angle. For each try we will have to generate d independent uniform $[0, 1]$ random real numbers

$$0 < r_1, r_2, \ldots, r_d < 1 \tag{25}$$

one random number per coordinate. Then we take coordinate i iff

$$r_i \leq \frac{p_i - p_{cut}}{1 - p_{cut}} \tag{26}$$

Let us reverse direction. Generate r_i first, and then compute for which p_{cut}'s coordinate i is taken:

$$p_{cut} \leq 2^{-\lambda_i} = \max\left[\frac{p_i - r_i}{1 - r_i}, 0\right] \tag{27}$$

Denoting the right hand side by $2^{-\lambda_i}$ is unnecessarily cumbersome at this stage, but will make sense later. We will call λ_i the **random exponent** of coordinate i (random because it is r_i dependent). Remember that $p_{cut} > 0$ so $\lambda_i = \infty$ means that for that value of r_i coordinate i can not be used. Now which value of p_{cut} will get us k coordinates? There is no need to solve equations. Sort the λ_i's in nondecreasing order, and pick out the first k. Hence

$$p_{cut} = 2^{-\lambda_{cut}} \tag{28}$$

where the **cutoff exponent** λ_{cut} is the value of the $k'th$ ordered random exponent.

It takes some time to comprehend the effect of equation (27). The random element seems overwhelming. The probability that coordinate 1 will have larger random exponent than coordinate 2 when $p_1 > p_2$ is

$$\frac{1}{2}\frac{1-p_1}{1-p_2} \tag{29}$$

In particular the probability that a useless coordinate with $p_i = 0.5$ precedes a good coordinate with $p_i = 0.9$ is 0.1 ! However the chance that the useless coordinate will be ranked among the first k is very small, unless we have so little data that it is better to take $k < \ln n_0$.

IV. An Unlimited Homogeneous Data Example

The previous section completely avoids an important aspect of the general problem which will be presented by the following example. Suppose we have an unlimited amount of data $d \to \infty$ of the same type

$$P = \begin{pmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{pmatrix} \qquad 1 \le i \le d \tag{30}$$

where

$$p_{00} + p_{01} + p_{10} + p_{11} = 1 \tag{31}$$

This is the joint probability of the dependent pair, and the marginal probabilities govern the distribution of the remaining points. In the set X_0 the probability that bit i is 0 is

$$p_{0*} = p_{00} + p_{01} \tag{32}$$

and similarly in X_1

$$p_{*0} = p_{00} + p_{10} \tag{33}$$

The * means “don’t care”. A reasonable pairing algorithm (very similar in this case to the general algorithm) is to pick coordinates at random $1 \le i_1, i_2, \ldots \le d$. After picking k coordinates, an X_0 point $x_l = (x_{l1}, x_{l2}, \ldots, x_{ld})$ is in a bucket of expected size

$$n_0 \prod_{t=1}^{k} p_{x_{lt}*} \tag{34}$$

Hence it makes sense to increase k only up to the point where $n_0 \prod_{t=1}^{k} p_{x_{lt}*} < 1$, and then compare with all X_1 points in its cell. This makes k point dependent. The expected number of comparisons in a single try is at most n_1. What is the approximate success probability?

Our initial estimate was the following. The probability that the special pair will agree in a single coordinate is $p_{00}+p_{11}$ The amount of information in a single X_0 coordinate is $-p_{0*}\ln p_{0*}-p_{1*}\ln p_{1*}$ so we will need about

$$k \approx \frac{\ln n_0}{-p_{0*}\ln p_{0*} - p_{1*}\ln p_{1*}} \tag{35}$$

coordinates, and the success probability is estimated by

$$(p_{00}+p_{11})^k \approx n_0^{-\frac{\ln(p_{00}+p_{11})}{p_{0*}\ln p_{0*}+p_{1*}\ln p_{1*}}} \tag{36}$$

This estimate turns out to be disastrously wrong. For the bad matrix

$$\begin{pmatrix} 1-2\epsilon & \epsilon \\ \epsilon & 0 \end{pmatrix} \tag{37}$$

with small ϵ it suggests exponent $-1/\ln\epsilon$, while clearly it is worse than 1. The interested reader might pause to figure out what went wrong, and how this argument can be salvaged.

There is an almost exact simple answer with a surprising flavor. We expect $n_0^{-\lambda}$, so let us check that for consistency. Pick the first coordinate. With probability p_{00}, the expectation n_0 is reduced to $n_0 p_{0*}$. With probability p_{11} it is reduced to $n_0 p_{1*}$, and with probability $p_{22} = 1 - p_{00} - p_{11}$ the try is already lost. Hence

$$n_0^{-\lambda} \approx p_{00}(n_0 p_{0*})^{-\lambda} + p_{11}(n_0 p_{1*})^{-\lambda} \tag{38}$$

Happily n_0 drops out, leaving us with

$$p_{00}p_{0*}^{-\lambda} + p_{11}p_{1*}^{-\lambda} = 1 \tag{39}$$

which determines the exponent λ. It is very easy to convert this informal argument into a formal theorem and proof. A harder task awaits us.

V. The General Algorithm and its Performance

Definition 5.1: The independent data model is the following. We generalize from bits to b discrete values. Let the sets

$$X_0, X_1 \subset \{0,1,\ldots,b-1\}^d \tag{40}$$

of cardinalities

$$\#X_0 = n_0, \quad \#X_1 = n_1 \tag{41}$$

be randomly constructed in the following way. The X_0 points are identically distributed independent Bernoulli random vectors, with $p_{i,j*}$ denoting the probability that coordinate i has value j. There is a special pair of X_0, X_1 points, randomly chosen out of the $n_0 n_1$ possibilities. For that pair the probability that both their i'th coordinates equal j is $p_{i,j}$ with no dependency between coordinates. The rest of the X_1 points can be anything. (We abbreviate the usual notation $p_{i,jj}$ to $p_{i,j}$, because we will consider only the diagonal and the marginal probabilities.) Denote

$$p_{i,b} = 1 - \sum_{j=0}^{b-1} p_{i,j} \tag{42}$$

$$P_i = \begin{pmatrix} p_{i,0} & p_{i,1} & \cdots & p_{i,b-1} \\ p_{i,0*} & p_{i,1*} & \cdots & p_{i,b-1\ *} \end{pmatrix} \tag{43}$$

We propose the following algorithm. It consists of several bucketing tries. For each try we generate d independent uniform $[0,1]$ random real numbers

$$0 < r_1, r_2, \ldots, r_d < 1 \tag{44}$$

one random number per coordinate. For each coordinate i we define its **random exponent** $\lambda_i \geq 0$ to be the unique solution of the monotone equation

$$\sum_{j=0}^{b-1} \frac{p_{i,j}}{(1-r_i)p_{i,j*}^{\lambda_i} + r_i} = 1 \tag{45}$$

or $+\infty$ when there is no solution. ($p_{i,j*}^{\lambda_i}$ means $(p_{i,j*})^{\lambda_i}$). We lexicographically sort all the $n_0 + n_1$ points, with lower exponent coordinates given precedence over larger exponent coordinates, and the coordinate values $0, 1, \ldots, b-1$ arbitrarily arranged, even without consistency. Now each X_1 point is compared with the preceding a and following a X_0 points (or fewer near the ends). The comparisons are done in some one-on-one way, and the algorithm is considered successful if it asks for the correct comparison. The best a is problem and computer dependent, but is never large. Each try makes at most $2an_1$ comparisons. Of course there is extra $n_0 + n_1$ point handling work.

A nice way to write the lexicographic ordering of the algorithm follows. Suppose that in try t the sorted random exponents are

$$\lambda_{\pi_1} < \lambda_{\pi_2} < \cdots < \lambda_{\pi_d} \tag{46}$$

Then each point

$$x = (x_1, x_2, \ldots, x_d) \in \{0, 1, \ldots, b-1\}^d \tag{47}$$

is projected into the interval $[0,1]$ by

$$R_t(x) = \sum_{i=1}^{d} p_{\pi_1,x_{\pi_1}*}\ p_{\pi_2,x_{\pi_2}*}\ \cdots\ p_{\pi_{i-1},x_{\pi_{i-1}}*} \sum_{j=0}^{x_{\pi_i}-1} p_{\pi_i,j*}$$

The projection order is a lexicographic order. For large dimension d, $R_t(x)$ is approximately uniformly distributed in $[0,1]$.

We will prove that the number of tries T needed for success satisfies

$$\ln T \sim \max_{\lambda \geq 0} \left[\lambda \ln n_0 - \sum_{i=1}^{d} F(P_i, \lambda) \right] \tag{48}$$

where

Definition 5.2: The bucketing forest information function $F(P_i,\lambda)$ is

$$F(P_i,\lambda) = \min_{\substack{0 \leq q_{i,0},\ldots,q_{i,b} \\ \sum_{j=0}^{b} q_{i,j} = 1 \\ \sum_{j=0}^{b-1} \frac{q_{i,j}}{p_{i,j*}^{\lambda}} \leq 1}} \sum_{j=0}^{b} p_{i,j} \ln \frac{p_{i,j}}{q_{i,j}} = \tag{49}$$

$$= \max_{0 \leq r_i \leq 1} \sum_{j=0}^{b} p_{i,j} \ln \left(1 - r_i + r_i \frac{(j \neq b)}{p_{i,j*}^{\lambda}} \right) \tag{50}$$

The two dual extrema points are related by

$$q_{i,j} = \frac{p_{i,j}}{1 - r_i + r_i \frac{(j \neq b)}{p_{i,j*}^{\lambda}}} \tag{51}$$

For $\sum_{j=0}^{b-1} \frac{p_{i,j}}{p_{i,j*}^{\lambda}} \leq 1$ $r_i = 0,\ q_{i,j} = p_{i,j},\ F(P_i,\lambda) = 0$. Otherwise

$$\sum_{j=0}^{b-1} \frac{q_{i,j}}{p_{i,j*}^{\lambda}} = \sum_{j=0}^{b-1} \frac{p_{i,j}}{(1-r_i)p_{i,j*}^{\lambda} + r_i} = 1 \tag{52}$$

We will get (49) from the upper bound theorem, and (50) from the lower bound theorem. Their equivalence is a simple (though a bit surprising) application of Lagrange multipliers in a convex setting. Representation (50) implies that $F(P,\lambda)$ is an increasing convex function of λ.

The **cutoff exponent** λ_{cut} attains (48). It has several meanings.

1) In each try the coordinates with $\lambda_i \leq \lambda_{cut}$ define a bucket of size $e^{\epsilon n_0}$ for some small real ϵ.
2) If we double n_0 the number of tries needed to achieve success probability $1/2$ is approximately multiplied by $2^{\lambda_{cut}}$.

3) If we delete coordinate i, then the number of tries needed to achieve success probability $1/2$ is on average multiplied by $e^{F_{i,p}(\lambda_{cut})}$.

Switching X_0 and X_1 may result in a different algorithm. Coordinate values can be changed and/or merged in possibly different ways for X_0, X_1. For each possibility we have an estimate of its effectiveness, and the best should be taken.

In real applications there is dependence, and the probabilities have to be estimated. Our practical experience indicates that this is a robust algorithm. Details will be described in another paper.

VI. An Alternative Algorithm

There is an interesting alternative to the random ordering of coordinates. Suppose we have training sets X_0, X_1 both of size n, such that each X_0 point is paired with a known X_1 point. Let us estimate the probabilities P_i by their empirical averages. For each coordinate i its **exponent** $\lambda_i \geq 0$ is defined by

$$\sum_{j=0}^{b-1} \frac{p_{i,j}}{p_{i,j*}^{\lambda_i}} = 1 \tag{53}$$

Arrange the coordinates in the **greedy order** of nondecreasing exponents. Perform the first try using that order just like in the previous algorithm. Remove the pairs found from the training data, and repeat recursively on the reduced training data. Stop after the training set is reduced to $1/3$ (for example) of its original size, or you run out of memory. The memory problem can be alleviated by keeping only the heads of coordinate lists, and/or running training and working tries in parallel.

This simpler algorithm has a more complicated and/or less efficient implementation, and lacks theory.

VII. Return of the Marginally Bernoulli(1/2) Example

For the marginally Bernoulli(1/2) example equation (45) is

$$2\frac{p_i/2}{(1-r_i)2^{-\lambda_i} + r_i} = 1 \tag{54}$$

which can be recast as the familiar

$$2^{-\lambda_i} = \frac{p_i - r_i}{1 - r_i} \tag{55}$$

The bucketing forest information function is

$$F(P_i,\lambda)=\begin{cases} p_i\ln\frac{p_i}{2^{-\lambda}}+(1-p_i)\ln\frac{1-p_i}{1-2^{-\lambda}} & p_i\geq 2^{-\lambda}\\ 0 & p_i\leq 2^{-\lambda}\end{cases}$$

The cutoff exponent attains (48). The extremal condition is the familiar

$$\sum_{i=1}^{d}\max\left[\frac{p_i-2^{-\lambda_{cut}}}{1-2^{-\lambda_{cut}}},0\right]=\log_2 n_0 \tag{56}$$

Let us now specialize to $p_1=p_2=\cdots=p_d=p$. Then

$$\lambda_{cut}=-\log_2\frac{pd-\log_2 n_0}{d-\log_2 n_0} \tag{57}$$

Notice that $\log_2 n_0>(2p-1)d$ is equivalent to $\lambda_{cut}>1$. In general $\lambda_{cut}>1$ signals that the available bucketing forest information is of such low quality that the trees are worse than random near their leafs.

VIII. Sparsity

Let us specialize to sparse bits: $b=2$,

$$p_{i,1*},p_{i,}+p_{i,11}<<1 \tag{58}$$

We will also assume that for some fixed $\delta>0$

$$p_{i,11}\geq\delta\left(p_{i,01}+p_{i,10}\right) \tag{59}$$

The equation

$$\frac{p_{i,00}}{(1-r_i)p_{i,0*}^{\lambda_i}+r_i}+\frac{p_{i,11}}{(1-r_i)p_{i,1*}^{\lambda_i}+r_i}=1 \tag{60}$$

has two asymptotic regimes: one in which $p_{i,0*}^{\lambda_i}$ is nearly constant and $p_{i,1*}^{\lambda_i}$ changes, and vice versa. The first regime is the important one:

$$p_{i,00}+\frac{p_{i,11}}{(1-r_i)p_{i,1*}^{\lambda_i}+r_i}\approx 1 \tag{61}$$

$$\lambda_i\approx\frac{\ln\left[1-\frac{1}{(1-r_i)\left(1+\frac{p_{i,11}}{p_{i,01}+p_{i,10}}\right)}\right]}{\ln p_{i,1*}} \tag{62}$$

In practice the probabilities have to be estimated from the data, and sparse estimates must be unreliable, so we used the more conservative

$$1/\tilde{\lambda}_i = (1 - r_i)\frac{p_{i,11}}{p_{i,01} + p_{i,10}} \ln \frac{1}{p_{i,1*}} \tag{63}$$

A very important practical point is that the general algorithm preserves sparsity. Suppose that instead of points

$$x = (x_1, x_2, \ldots, x_d) \in \{0, 1\}^d \tag{64}$$

we have subsets of a features set D of cardinality d :

$$D_x \subset D \tag{65}$$

In try t we use a hash function $\mathrm{hash}_t : D \to [0, 1]$. For each feature $i \in D$ its random exponent λ_i is computed using the pseudo random

$$r_i = \mathrm{hash}_t(i) \tag{66}$$

and the random exponents of x are sorted

$$\lambda_{\pi_1} < \lambda_{\pi_2} < \cdots < \lambda_{\pi_\nu} \tag{67}$$

Then the sequence of features

$$(\pi_1, \pi_2, \ldots, \pi_\nu) \tag{68}$$

is a sparse representation of x whose lexicographic order is used in try t.

IX. The Downside of Dimensionality Reduction

Another way of handling sparse approximate neighbor problems is to convert them into dense problems by a random projection. For dense problems taking some k out of the d coordinates can be an effective way to reduce dimension. For sparse problems such a sampling reduction will remain sparse, hence dense projection matrices are used instead. We will show that this can result in a much worse algorithm. Let us consider the unlimited homogeneous data example with

$$p_{01} = p_{10}, \quad n_0 = n_1 = n \tag{69}$$

because in general it is not clear which projections to take and how to analyze their performance. We have a d dimensional Hamming cube $\{0,1\}^d$. The Hamming distance between two random X_0, X_1 points is approximately

$$2p_{0*}p_{1*}d \tag{70}$$

The Hamming distance between the two special points is approximately

$$2p_{0,1}d \tag{71}$$

Hence when the dimension d is large, the random to special distances ratio tends to

$$c = \frac{p_{0*}p_{1*}}{p_{01}} \tag{72}$$

The ideal dimensionality reduction would be to project $\{0,1\}^d$ into a much lower dimensional $\{0,1\}^k$ in such a way that the images of the X_0, X_1 points are random $\{0,1\}^k$ points, and the distance between the two special images is approximately $k/2c$ ($k/2$ is the approximate distance between two random image points). Hence after the dimensionality reduction we will have a homogeneous marginally Bernoulli(1/2) problem with

$$p = 1 - 1/2c \tag{73}$$

The standard nearest neighbor algorithm solves this in approximately

$$n^{\log_2 \frac{2c}{2c-1}} \tag{74}$$

tries. Actual dimensional reductions fall short of this ideal. The Indyk and Motwani theory [5] states that

$$n^{1/c} \tag{75}$$

tries suffice. The truth is somewhere in between.

In contrast without dimensionality reduction our algorithm takes approximately n^λ tries where λ is determined by

$$\frac{1 - p_{1*} - p_{01}}{(1-p_{1*})^\lambda} + \frac{p_{11}}{p_{1*}^\lambda} = 1 \tag{76}$$

In the asymptotic region (58,59) inserting $r = 0$ into (62) results in

$$\lambda \approx \frac{\ln\left[1 + \frac{2p_{01}}{p_{11}}\right]}{\ln 1/p_{1*}} \approx \frac{\ln \frac{c+1}{c-1}}{\ln 1/p_{1*}} \tag{77}$$

We encourage the interested reader to look at his favorite dimensional reduction scheme, and see that the $\ln 1/p_{1*}$ factor is really lost.

X. LEXICOGRAPHIC AND BUCKETING FORESTS

Our general algorithm is of the following type.

Definition 10.1: A lexicographic tree algorithm is the following. The d coordinates are arranged according to some permutation. Than a complete lexicographic ordered tree is generated. It is defined recursively as a root pointing towards b subtrees, with the edges denoting the possible values of the first (after permutation) coordinate arbitrarily ordered. The subtrees are complete lexicographic ordered trees for the remaining $d-1$ coordinates. In particular the lexicographic tree has b^d ordered leafs, each denoting a point in $\{0,1,\ldots,b-1\}^d$. A lexicographic tree algorithm arranges the n_0+n_1 $X_0 \cup X_1$ points according to the tree, and then compares each x_1 point with its a neighbors right and left. This insures no more than $2an_1$ comparisons per tree. A lexicographic forest is simply a forest of lexicographic trees, each having its own permutation. It succeeds iff at least one tree succeeds.

An obvious generalization is

Definition 10.2: A semi-lexicographic tree algorithm has a 'first' coordinate and then recursively each subtree is semi-lexicographic, until all coordinates are exhausted.

For example we can start with coordinate 3, and than consider coordinate 5 if the value is 0, or coordinate 2 if the value is 1 and so on.

The success probability of a lexicographic forest is very complicated,even before randomizing the algorithm. For that reason we will consider an uglier non-robust class of algorithms that are easier to understand and analyze.

Definition 10.3: A bucketing tree algorithm is predictably recursively defined. Either compare all pairs (a leaf bucket), or take one coordinate, split the data into b parts according to its value (some parts may be empty), and apply a bucketing tree algorithm on each part separately. In order to have no more than an_0 expected comparisons we will insist that each leaf expects no more than a points belonging to X_0. A bucketing forest is simply a forest of bucketing trees. It succeeds iff at least one tree succeeds.

The success probability of a bucketing forest is no bed of roses. Let us denote a leaf by $w \in \{0,1,\ldots,b\}^d$, with b indicating that the corresponding coordinate is not taken. The leaf w expects

$$n_0 \prod_{i=1}^{d} \begin{cases} p_{i,w_i*} & w_i < b \\ 1 & w_i = b \end{cases} \tag{78}$$

X_0 points, and its success probability is

$$\prod_{i=1}^{d} \begin{cases} p_{i,w_i} & w_i < b \\ 1 & w_i = b \end{cases} \tag{79}$$

The success probability of a tree is the sum of the success probabilities of its leafs. The success probability of the whole forest is less than the tree sum. Suppose the whole forest contains L leafs $w_1, w_2, \ldots, w_L$. Let $y \in \{0, 1, \ldots, b\}^d$ denote the abbreviated state of the special points:

$$y_i = \begin{cases} x_{0,i} & x_{0,i} = x_{1,i} \\ b & x_{0,i} \neq x_{1,i} \end{cases} \tag{80}$$

The value b denotes disagreement and its probability is $p_{i,b} = 1 - \sum_{j=0}^{b-1} p_{i,j}$. The success probability of the whole forest is

$$S = \sum_{y \in \{0,1,\ldots,b\}^d} \prod_{i=1}^{d} p_{i,y_i} \cdot \cdot \left[1 - \prod_{l=1}^{L} \left(1 - \prod_{i=1}^{d} (w_{l,i} == y_i \;||\; w_{l,i} == b) \right) \right]$$

Remember that $(w_{l,i} == y_i \;||\; w_{l,i} == b) = 0, 1$ hence the two rightmost products are just logical ands, and $1 - ()$ is a logical not.

Our algorithm is almost a bucketing forest, except that the leaf condition is data dependent (for robustness). A truly variable scheme can shape the buckets in a more complicated data dependent way, see for example Gennaro Savino and Zezula [4]. Non-tree bucketing can use several coordinates together, so that the resulting buckets are not boxes, see for example Andoni and Indyk [2] or [7].

XI. A Bucketing Forest Upper Bound

In this section we will bound the performance of bucketing forest algorithms. It is tricky, but technically simpler and more elegant than proving a lower bound on the performance of a single algorithm.

Theorem 11.1: Assume the independent data model. The success probability P of a nonempty bucketing tree whose leafs all have probabilities at most $1/N$ is at most

$$P \leq N^{-\lambda} \prod_{i=1}^{d} \max \left(1, \sum_{j=0}^{b-1} \frac{p_{i,j}}{p_{i,j*}^{\lambda}} \right) \tag{81}$$

for any $\lambda \geq 0$. We do not even have to assume $p_{i,j} \leq p_{i,j*}$.

Proof: Use induction. Without losing generality split coordinate 1. The induction step

$$P \leq \sum_{j=0}^{b-1} p_{1,j} \left(N p_{1,j*}\right)^{-\lambda} \prod_{i=2}^{d} \max\left(1, \sum_{j=0}^{b-1} \frac{p_{i,j}}{p_{i,j*}^{\lambda}}\right) \tag{82}$$

is valid for both proper and point-only subtrees. The maximization with 1 is necessary because coordinates can be ignored. ■

Theorem 11.2: Assume the independent data model. Suppose an bucketing forest contains T trees, its success probability is S, and all its leafs have probabilities at most $1/N$. Than for any $\lambda \geq 0$

$$\ln T \geq \lambda \ln N + \ln \frac{S}{2} - \sqrt{\frac{4}{S} \sum_{i=1}^{d} V(P_i, \lambda)} - \sum_{i=1}^{d} F(P_i, \lambda)$$

where

$$V(P_i, \lambda) = \sum_{j=0}^{b} p_{i,j} \left(\ln \frac{p_{i,j}}{q_{i,j}} - \sum_{k=0}^{b} p_{i,k} \ln \frac{p_{i,k}}{q_{i,k}}\right)^2 \tag{83}$$

and the $q_{i,j}$'s are the minimizing arguments from F's definition (49)

Proof: The previous theorem provides a good bound for the success probability of a single tree, but it is not tight for a forest, because of dependence: the failure of each tree increases the failure probability of other trees. Now comes an interesting argument. Recall that the success probability of the whole forest formula (81). For any z and $q_{i,j} > 0$ we can bound

$$S \leq \mathrm{Prob}\{Z \geq z\} + e^z S_Q \tag{84}$$

where

$$Z = \sum_{i=1}^{d} \ln \frac{p_{i,y_i}}{q_{i,y_i}} \tag{85}$$

$$\mathrm{Prob}\{Z \geq z\} = \sum_{y \in \{0,1,\ldots,b\}^d} \prod_{i=1}^{d} p_{i,y_i} \cdot \left(\sum_{i=1}^{d} \ln \frac{p_{i,y_i}}{q_{i,y_i}} \geq z\right)$$

$$S_Q = \sum_{y \in \{0,1,\ldots,b\}^d} \prod_{i=1}^{d} q_{i,y_i} \cdot \cdot \left[1 - \prod_{l=1}^{L} \left(1 - \prod_{i=1}^{d} (w_{l,i} == y_i \;||\; w_{l,i} == b)\right)\right]$$

We insist upon

$$\sum_{j=0}^{b} q_{i,j} = 1 \tag{86}$$

so that we can use the previous lemma to bound

$$S_Q \leq TP_q \leq TN^{-\lambda} \prod_{i=1}^{d} \max\left(1, \sum_{j=0}^{b-1} \frac{q_{i,j}}{p_{i,j*}^{\lambda}}\right) \tag{87}$$

The other term is handled by the Chebyshev bound: for $z > \mathrm{E}(Z)$

$$\mathrm{Prob}\{Z \geq z\} \leq \frac{\mathrm{Var}(Z)}{(z - \mathrm{E}(Z))^2} \tag{88}$$

Together

$$S \leq \frac{\mathrm{Var}(Z)}{(z - \mathrm{E}(Z))^2} + e^z S_Q \tag{89}$$

The reasonable choice of

$$z = \mathrm{E}(Z) + \sqrt{2\mathrm{Var}(Z)/S} \tag{90}$$

results in

$$S \leq 2e^{\mathrm{E}(Z) + \sqrt{2\mathrm{Var}(Z)/S}} S_Q \tag{91}$$

■

Notice that this proof gives no indication that the bound is tight, nor guidance towards constructing an actual bucketing forest, (except for telling which coordinates to throw away).

We tried to strengthen the theorem in the following way. Instead of restricting the expected number of points falling into each leaf bucket, allow larger leafs and only insist that the total number of comparisons is at most aN. Surprisingly the strengthened statement is wrong, and a 'large leafs' bucketing forest is theoretically better than our algorithm. But it is complicated and non-robust.

XII. A Semi-Lexicographic Forest Upper Bound

There remains the problem that we gave a lexicographic forest algorithm, but a bucketing forest upper bound. It is a technicality, which may be skipped over with little loss. Any semi-lexicographic complete tree can be converted into a bucketing tree in an obvious way: Prune the complete tree from the leafs down as much as possible, preserving the property that each leaf expects at most $a/2$ points from X_0. The success probability of the semi-lexicographic tree is bounded by

$$P \leq P_{tree} + R \tag{92}$$

where P_{tree} is the success probability of the truncated tree, for which we have a good bound, and a remainder term associated with truncated tree vertexes expecting more than $a/2$ tree points.

Lemma 12.1: Assume the independent data model and consider a semi-lexicographic tree with the standard coordinate order (that does not restrict generality) and a totally random values order. Assume that the special points pair agree in coordinates $1, 2, \ldots, i-1$, but disagree at coordinate i :

$$y_1, y_2, \ldots, y_{i-1} \neq b, \ y_i = b \tag{93}$$

Conditioning on that, the probability of success is at most

$$\frac{2a}{n_0 p_{1,y1} p_{2,y_2} \cdots p_{i-1,y_{i-1}}} \tag{94}$$

Proof: Denote

$$p = p_{1,y1} p_{2,y_2} \cdots p_{i-1,y_{i-1}} \tag{95}$$

Let m be the number of X_0 points agreeing with the special pair in their first $i-1$ coordinates. Its probability distribution is 1+Bernoulli($p, n_0 - 1$). Let us consider these m points ordered by the algorithm. The rank of the special X_0 point can be $1, 2, \ldots, m$ with equal probabilities. Those m ordered points are broken up into up to b intervals according to the value of coordinate i. Where does the special X_1 point fit in? It is in a different interval than the X_0 special point, but its location in that interval, and the order of intervals is random. Hence the probability that the two special points are at most $a+1$ apart is at most $2a/m$. This has to be averaged:

$$\sum_{m=1}^{n} \left(\begin{array}{c} n-1 \\ m-1 \end{array} \right) p^{m-1} (1-p)^{n-m} \frac{2a}{m} = \frac{2a}{np} \tag{96}$$

■

Theorem 12.2: Assume the independent data model. Then the success probability of any semi-lexicographic tree with a totally random coordinate values order is at most

$$P \leq \frac{2 \ln (e^{4.5} N)}{N^{\lambda}} \prod_{i=1}^{d} \max \left(1, \sum_{j=0}^{b-1} \frac{p_{i,j}}{p_{i,j*}^{\lambda}} \right) \tag{97}$$

for any $0 \leq \lambda \leq 1$, where

$$N = \max \left(1, \frac{2n_0}{a} \right) \tag{98}$$

Proof: Without restricting generality assume that the coordinate have the standard order. We have established that

$$R \leq \sum_{\substack{0 \leq t \leq d \\ 0 \leq w_1, w_2, \ldots, w_t < b \\ N \prod_{i=1}^{t} p_{i,w_i*} \geq 1}} \frac{4}{N} \prod_{i=1}^{t} \frac{p_{i,w_i}}{p_{i,w_i*}} \cdot \left(1 - \sum_{j=0}^{b-1} p_{t+1,j}\right)$$

The negative terms can be shifted to the next t :

$$R \leq \frac{4}{N} + \sum_{\substack{1 \leq t \leq d \\ 0 \leq w_1, w_2, \ldots, w_t < b \\ N \prod_{i=1}^{t} p_{i,w_i*} \geq 1}} \frac{4}{N} \prod_{i=1}^{t} \frac{p_{i,w_i}}{p_{i,w_i*}} \cdot (1 - p_{t,w_t*})$$

Denote

$$\tilde{R}_{w_1,\ldots,w_s} = \sum_{\substack{s \leq t \leq d \\ 0 \leq w_{s+1}, w_{s+2}, \ldots, w_t < b \\ N \prod_{i=1}^{t} p_{i,w_i*} \geq 1}} \prod_{i=s+1}^{t} \frac{p_{i,w_i}}{p_{i,w_i*}} \cdot (1 - p_{t,w_t*})$$

We will prove by induction from the leafs down that

$$\tilde{R}_{w_1,w_2,\ldots,w_s} \leq N_{w_1,\ldots,w_s}^{1-\lambda} \ln\left(eN_{w_1,\ldots,w_{s-1}}\right) \cdot \tag{99}$$

$$\cdot \prod_{i=s+1}^{d} \max\left(1, \sum_{j=0}^{b-1} \frac{p_{i,j}}{p_{i,j*}^{\lambda}}\right) \tag{100}$$

where

$$N_{w_1,\ldots,w_s} = N \prod_{i=1}^{s} p_{i,w_i*} \tag{101}$$

The induction step boils down to

$$\ln\left(eN_{w_1,\ldots,w_{s-1}}\right) \geq (1 - p_{s,w_s*}) + \ln\left(eN_{w_1,\ldots,w_{s-1}} p_{s,w_s*}\right)$$

which is obviously true. ■

Theorem (11.2) is converted into

Theorem 12.3: Assume the independent data model. Suppose a semi-lexicographic forest with a totally random coordinate values order contains T trees, its success probability is S, and

$$N = \max\left(1, \frac{2n_0}{a}\right) \tag{102}$$

Than for any $0 \leq \lambda \leq 1$

$$\ln T \geq \lambda \ln N - \ln \left[2 \ln \left(e^{4.5} N \right) \right] + \tag{103}$$

$$+ \ln \frac{S}{2} - \sqrt{\frac{4}{S} \sum_{i=1}^{d} V(P_i, \lambda)} - \sum_{i=1}^{d} F(P_i, \lambda) \tag{104}$$

XIII. A Lower Bound

Theorem 13.1: Assume the independent data model and denote

$$N = \frac{2n_0}{a} \tag{105}$$

Let $\epsilon > 0$ be some small parameter, and let Let $\lambda, r_1, r_2, \ldots, r_d$ attain

$$\min_{\lambda \geq 0} \max_{0 \leq r_1, \ldots, r_d \leq 1} \left[- (1+\epsilon) \lambda \ln N + \right. \tag{106}$$

$$\left. + \sum_{i=1}^{d} \sum_{j=0}^{b} p_{i,j} \left(1 - r_i + r_i \frac{(j \neq b)}{p_{i,j*}^{\lambda}} \right) \right] \tag{107}$$

The extrema conditions are

$$\sum_{i=1}^{d} \sum_{j=0}^{b-1} p_{i,j} \frac{-r_i \ln p_{i,j*}}{(1-r_i) p_{i,j*}^{\lambda} + r_i} = (1+\epsilon) \ln N \tag{108}$$

and $r_i = 0$ or

$$\sum_{j=0}^{b-1} \frac{p_{i,j}}{(1-r_i) p_{i,j*}^{\lambda} + r_i} = 1 \qquad 1 \leq i \leq d \tag{109}$$

Suppose that for some $\delta < 1/7$

$$\sum_{i=1}^{d} \sum_{j=0}^{b} p_{i,j} \left(\ln[1 - r_i + (j \neq b) r_i p_{i,j*}^{-\lambda}] - \right.$$

$$\left. - \sum_{k=0}^{b} p_{i,k} \ln[1 - r_i + (k \neq b) r_i p_{i,k*}^{-\lambda}] \right)^2 \leq \epsilon^2 \delta \lambda^2 (\ln N)^2$$

$$\sum_{i=1}^{d} \sum_{j=0}^{b-1} p_{i,j} \left(\frac{-r_i \ln p_{i,j*}}{(1-r_i) p_{i,j*}^{\lambda} + r_i} - \right.$$

$$\left. - \sum_{k=0}^{b-1} p_{i,k} \frac{-r_i \ln p_{i,k*}}{(1-r_i) p_{i,k*}^{\lambda} + r_i} \right)^2 \leq \epsilon^2 \delta (\ln N)^2 / 4$$

$$\sum_{i=1}^{d} \sum_{j=0}^{b-1} p_{i,j} \frac{r_i (1-r_i) [\ln p_{i,j*}]^2}{\left[(1-r_i) p_{i,j*}^{\lambda} + r_i \right]^2} \leq \epsilon^2 \delta (\ln N)^2 / 8 \tag{110}$$

Then the general algorithm with T tries where

$$\ln T \geq \ln \frac{1}{\delta} + (1+3\epsilon)\lambda \ln N - \tag{111}$$

$$- \sum_{i=1}^{d} \sum_{j=0}^{b} p_{i,j} \left(1 - r_i + r_i \frac{(j \neq b)}{p_{i,j*}^{\lambda}} \right) \tag{112}$$

has success probability

$$S \geq 1 - 7\delta \tag{113}$$

Moreover there exists a bucketing forest with T trees and at least $1 - 7\delta$ success probability.

The alarmingly complicated small variance conditions are asymptotically valid, because the variances grow linearly with $\ln N$. However there is no guarantee that they can be always met. Indeed the upper bound is of the Chernof inequality large deviation type, and can be a poor estimate in pathological cases.

Definition 13.1: Let Y, Z be joint random variables. We denote by Y_Z the conditional type random variable Y with its probability density multiplied by

$$\frac{e^Z}{\mathrm{E}[e^Z]} \tag{114}$$

In the discrete case Z, Y would have values y_i, z_i with probability p_i. Then Y_Z has values y_i with probability

$$\frac{p_i e^{z_i}}{\sum_j p_j e^{z_j}} \tag{115}$$

Lemma 13.2: For any random variable Z, and $\lambda \geq 0$

$$\ln \mathrm{Prob}\left\{ Z \geq \mathrm{E}\left[Z_{\lambda Z} \right] \right\} \leq \ln \mathrm{E}\left[e^{\lambda Z} \right] - \lambda \mathrm{E}\left[Z_{\lambda Z} \right] \tag{116}$$

$$\ln \mathrm{Prob}\left\{ Z \geq \mathrm{E}\left[Z_{\lambda Z} \right] - \sqrt{2\mathrm{Var}\left[Z_{\lambda Z} \right]} \right\} \geq \tag{117}$$

$$\geq \ln \mathrm{E}\left[e^{\lambda Z} \right] - \lambda \mathrm{E}\left[Z_{\lambda Z} \right] - \ln 2 - \lambda \sqrt{2\mathrm{Var}\left[Z_{\lambda Z} \right]} \tag{118}$$

Proof: The upper bound is the Chernof bound. The lower bound combines the Chebyshev inequality

$$\mathrm{Prob}\left\{ |Z_{\lambda Z} - \mathrm{E}[Z_{\lambda Z}]| \leq \sqrt{2\mathrm{Var}[Z_{\lambda Z}]} \right\} \geq \frac{1}{2} \tag{119}$$

with the fact that the condition in the curly bracket bounds the densities ratio:

$$\ln \frac{e^{\lambda Z}}{\mathrm{E}\left[e^{\lambda Z} \right]} = \ln \frac{e^{\lambda Z_{\lambda Z}}}{\mathrm{E}\left[e^{\lambda Z} \right]} \leq \tag{120}$$

$$\leq - \ln \mathrm{E}\left[e^{\lambda Z} \right] + \lambda \mathrm{E}\left[Z_{\lambda Z} \right] + \lambda \sqrt{2\mathrm{Var}\left[Z_{\lambda Z} \right]} \tag{121}$$

■

It is amusing, and sometimes useful to note that

$$\mathrm{E}[Z_{\lambda Z}] = \frac{\partial \ln \mathrm{E}\left[e^{\lambda Z}\right]}{\partial \lambda} \tag{122}$$

$$\mathrm{Var}[Z_{\lambda Z}] = \frac{\partial^2 \ln \mathrm{E}[e^{\lambda Z}]}{\partial \lambda^2} \tag{123}$$

We will now prove the theorem 13.1. *Proof:* Let $\lambda \geq 0$ be a parameter to be optimized. Let $w \in \{0,1\}^d$ be the random Bernoulli vector

$$w_i = (\lambda_i \leq \lambda) \tag{124}$$

where λ_i is the i'th random exponent. In a slight abuse of notation let $0 \leq r_i \leq 1$ denote not a random variable but a probability

$$r_i = \mathrm{Prob}\{w_i == 1\} = \mathrm{Prob}\{\lambda_i \leq \lambda\} \tag{125}$$

We could not resist doing that because equation (45) is still valid under this interpretation. Another point of view is to forget (45) and consider r_i a parameter to be optimized. Again let $y \in \{0,1,\ldots,b\}^d$ denote the abbreviated state of the special points x_0, x_1. Let us consider a single try of our algorithm, conditioned on both y and w. The following requirements

$$\prod_{i=1}^{d}(1 - w_i + w_i(y_i \neq b)) = 1 \tag{126}$$

$$\prod_{i=1}^{d}(1 - w_i + w_i p_{i,y_i*}) \leq \frac{1}{N} = \frac{a}{2n_0} \tag{127}$$

state that the expected number of X_0 points in the bucket defined by the coordinates whose $w_i = 1$ with value y_i is at most $a/2$. Then the probability that the actual number of bucket points is more than a is bounded from above by $1/2$. A more compact way of stating (126) and (127) together is

$$Z(y,w) \geq \ln N \tag{128}$$

$$Z(y,w) = \sum_{i=1}^{d} \ln\left[1 - w_i + w_i(y_i \neq b)p_{i,y_i*}^{-1}\right] \tag{129}$$

Summing over w gives success probability of a single try, conditioned over y to be at least

$$P(y) \geq \frac{1}{2} \sum_{w \in \{0,1\}^d} \prod_{i=1}^{d} [(1-w_i)(1-r_i) + w_i r_i][Z(y,w) \geq \ln N]$$

In short

$$P(y) \geq \frac{1}{2}\mathrm{Prob}\left\{Z(y) \geq \ln N\right\} \tag{130}$$

Conditioning over y makes tries independent of each other, hence the conditional success probability of at least T tries is at least

$$S(y) \geq 1 - (1 - P(y))^T \geq \frac{TP(y)}{1 + TP(y)} \tag{131}$$

Averaging over y bounds the success probability S of the algorithm by

$$S \geq \sum_{y \in \{0,1,\ldots,b\}^d} \prod_{i=1}^{d} p_{i,y_i} \cdot \left[\frac{TP(y)}{1 + TP(y)}\right] \tag{132}$$

In short

$$S \geq \mathrm{E}\left[\frac{TP(y)}{1 + TP(y)}\right] \tag{133}$$

Now we must get our hands dirty. The reverse Chernof inequality is

$$\ln \mathrm{Prob}\left\{Z(y) \geq \mathrm{E}\left[Z(y)_{\lambda Z(y)}\right] - \sqrt{2\mathrm{Var}\left[Z(y)_{\lambda Z(y)}\right]}\right\} \geq$$
$$\geq \ln \mathrm{E}\left[e^{\lambda Z(y)}\right] - \lambda \mathrm{E}\left[Z(y)_{\lambda Z(y)}\right] - \ln 2 -$$
$$-\lambda\sqrt{2\mathrm{Var}\left[Z(y)_{\lambda Z(y)}\right]}$$

Denoting

$$U(y) = \ln \mathrm{E}\left[e^{\lambda Z(y)}\right] = \sum_{i=1}^{d} \ln[1 - r_i + (y_i \neq b) r_i p_{i,y_i*}^{-\lambda}]$$

$$V(y) = \frac{\partial U(y)}{\partial \lambda} = \mathrm{E}\left[Z(y)_{\lambda Z(y)}\right] = \tag{134}$$

$$= \sum_{\substack{1 \leq i \leq d \\ y_i \neq b}} \frac{-r_i \ln p_{i,y_i*}}{(1 - r_i) p_{i,y_i*}^{\lambda} + r_i} \tag{135}$$

$$W(y) = \frac{\partial^2 U(y)}{\partial \lambda^2} = \mathrm{Var}\left[Z(y)_{\lambda Z(y)}\right] = \tag{136}$$

$$= \sum_{\substack{1 \leq i \leq d \\ y_i \neq b}} \frac{r_i(1 - r_i)[\ln p_{i,y_i*}]^2}{\left[(1 - r_i) p_{i,y_i*}^{\lambda} + r_i\right]^2} \tag{137}$$

the reverse Chernof inequality can be rewritten as

$$\ln \mathrm{Prob}\left\{Z(y) \geq V(y) - \sqrt{2W(y)}\right\} \geq \tag{138}$$

$$\geq U(y) - \lambda V(y) - \ln 2 - \lambda\sqrt{2W(y)} \tag{139}$$

It is time for the second inequality tier. For any $\delta < 1/3$

$$\mathrm{Prob}\Big\{|U(y) - \mathrm{E}[U]| \leq \sqrt{\mathrm{Var}[U]/\delta}, \tag{140}$$

$$|V(y) - \mathrm{E}[V]| \leq \sqrt{\mathrm{Var}[V]/\delta}, \tag{141}$$

$$W(y) \leq \mathrm{E}[W]/\delta\Big\} \geq 1 - 3\delta \tag{142}$$

where

$$\mathrm{E}[U] = \sum_{i=1}^{d}\sum_{j=0}^{b} p_{i,j} \ln[1 - r_i + (j \neq b) r_i p_{i,j*}^{-\lambda}] \tag{143}$$

$$\mathrm{E}[V] = \sum_{i=1}^{d}\sum_{j=0}^{b-1} p_{i,j} \frac{-r_i \ln p_{i,j*}}{(1-r_i)p_{i,j*}^{\lambda} + r_i} \tag{144}$$

Hence

$$\ln \mathrm{Prob}\Big\{Z(y) \geq \mathrm{E}[V] - \sqrt{\mathrm{Var}[V]/\delta} - \sqrt{2\mathrm{E}[W]/\delta}\Big\} \geq$$

$$\geq \mathrm{E}[U] - \lambda \mathrm{E}[V] - \ln 2 - \sqrt{\mathrm{Var}[U]/\delta} -$$

$$-\lambda\sqrt{\mathrm{Var}[V]/\delta} - \lambda\sqrt{2\mathrm{E}[W]/\delta}$$

Now we have to pull all strings together. In order to connect with (130) we will require

$$\mathrm{E}[V] = (1+\epsilon)\ln N \tag{145}$$

$$\sqrt{\mathrm{Var}[V]} + \sqrt{2\mathrm{E}[W]} \leq \epsilon\delta^{1/2}\ln N \tag{146}$$

for some small $\epsilon > 0$. Recalling (135), condition (145) is achieved by choosing λ to attain

$$\min_{\lambda \geq 0} [-(1+\epsilon)\lambda \ln N + \mathrm{E}[U]] \tag{147}$$

If (146) holds, then

$$\ln P(y) \geq -(1+2\epsilon)\lambda \ln N + \mathrm{E}[U] - \ln 4 - \sqrt{\mathrm{Var}[U]/\delta} \tag{148}$$

with probability at least $1 - 3\delta$. Recalling (133) the success probability is at least

$$S \geq \frac{1-3\delta}{1 + 4e^{(1+2\epsilon)\lambda \ln N - \mathrm{E}[U] + \sqrt{\mathrm{Var}[U]/\delta}}/T} \tag{149}$$

■

XIV. Conclusion

To sum up, we present three things:

1) An approximate nearest neighbor algorithm (45), and its sparse approximation (63).
2) An information style performance estimate (48).
3) A warning against dimensional reduction of sparse data, see section IX.

References

[1] N. Alon Private Communication.

[2] A. Andoni, P. Indyk *Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions* FOCS 2006.

[3] A. Broder. *Identifying and Filtering Near-Duplicate Documents* Proc. FUN, 1998.

[4] C.Gennaro, P.Savino and P.Zezula *Similarity Search in Metric Databases through Hashing* Proc. ACM workshop on multimedia, 2001.

[5] P. Indyk and R. Motwani. *Approximate Nearest Neighbor: Towards Removing the Curse of Dimensionality* Proc. 30th Annu. ACM Sympos. Theory Comput., 1998.

[6] R.M. Karp, O. Waarts, and G. Zweig. *The Bit Vector Intersection Problem* Proc. 36th Annu. IEEE Sympos. Foundations of Computer Science, pp. 621-630, 1995.

[7] *Bucketing Information and the Statistical High Dimensional Nearest Neighbor Problem* To be Published.